VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe
Individual Professional Practice in the Company

2019 Michal Borski

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michal Borski**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Tieto Czech s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Konzultant bakalářské práce: Ing. Roman Šimeček

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: 7.4.2019 .....

Podpis studenta

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této bakalářské/diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava."

V Ostravě dne: 24.4.2019

Tieto Czech s.r.o.
28. října 3346/91
702 00 Ostrava - Moravská Ostrava
IČO 64608051 DIC CZ64608051

Podpis zástupce

Poděkování

Rád bych poděkoval panu Ing. Petru Rajskému za přijetí do týmu ve firmě Tieto s.r.o., dále panu Ing. Romanu Šimečkovi za odborné rady a dohled při mé praxi a v neposlední řadě vedoucímu diplomové práce Ing. Davidu Ježkovi za odbornou pomoc a konzultaci při vytváření této práce.

Abstrakt

Tato bakalářská práce popisuje průběh mé odborné praxe, probíhající na pozici Software Developer ve firmě Tieto. V první části přestavuji firmu Tieto, mé pracovní zařazení a popis projektů, kterých jsem se účastnil. V druhé části je krátké seznámení se zadanými úkoly v průběhu praxe a jejich časová náročností. Ve třetí části se nachází detailní postup řešení jednotlivých úkolů. V závěru uvádím znalosti a dovednosti, které jsem v průběhu praxe využil, nebo mi naopak scházely a závěrečné zhodnocení celé praxe.

Klíčová slova

Tieto, odborná praxe, C#, .NET, Censhare, Censhare UI konfigurace

Abstract

This bachelor's thesis describes my internship in Tieto Czech. At the beginning, I briefly present Tieto, the job classification and projects I have worked on. Second part contains brief overview tasks, which I worked on and their time estimation. In third part I describe in detail the proposed solution of assigned tasks. In the end, I evaluate the overall course of the internship, the knowledge I gained by studying at the university and the knowledge which I had to study by myself.

Key words

Tieto, professional practise, C#, .NET, Censhare, Censhare UI configuration

Obsah

Seznam použitých symbolů a zkratek

.NET	Soubor technologií pro vývoj software
API	Aplikační rozhraní
ASP.NET	Součást .NET Frameworku pro tvorbu webových aplikací
CEM	Oddělení firmy Tieto
CSV	Soubor hodnot oddělených čárkami
ECMS	Technologie pro správu a řízení obsahu
EDMX	Soubor s databázovým modelem
HTML	Značkovací jazyk pro tvorbu www stánek
IT	Informační technologie
LINQ	Language Integrated Query
MDM	Správa kmenových dat
MVC	Model View Controller
ORM	Objektově relační mapování
OX	Pracovní skupina vytvářející komplexní řešení pro zákazníka
PIM	Řízení informací o produktech
QR	Rychlá odezva
REST	Architektura rozhraní, navržená pro distribuované prostředí
SQL	Structured Query Language
UI	Uživatelské rozhraní
UML	Graficky jazyk v softwarovém inženýrství
URL	Lokace zdrojů
UTF-8	Typ kódování
XML	Rozšiřitelný značkovací jazyk
XSL	Rozšiřitelný jazyk

Seznam tabulek

Seznam obrázků

Seznam ukázek zdrojových kódu

1 Úvod

Cílem této bakalářské práce je shrnout průběh odborné praxe, kterou jsem vykonával ve firmě Tieto Czech s.r.o. od začátku října 2018. Ve chvíli, kdy jsem se dozvěděl, že máme možnost absolvovat odbornou praxi ve firmě, místo klasické bakalářské práce, ani chvíli jsem nepřemýšlel nad tím, kterou možnost si vybrat. Konečně jsem měl možnost nabyté vědomosti využít v praxi, sáhnout si na reálné projekty pro reálné zákazníky, zdokonalit se v anglickém jazyce a v komunikaci a prakticky si vyzkoušet věci a situace, které škola většinou nabízí jen teoreticky.

Nejprve představím firmu Tieto, poté postupně popíši všechny úkoly, na kterých jsem pracoval, jejich časovou náročnost a podrobný postup jejich řešení. V rámci své praxe jsem zažil spoustu situací, které nejsou pro studentské stáže obvyklé. Jednalo se zejména o velmi častou komunikaci se zákazníkem ohledně jeho požadavků v anglickém jazyce, verifikační schůzky s prezentací vykonané práce před zákazníkem, a nakonec služební cestu do Mnichova, sídla společnosti Censhare, kde jsme absolvovali workshop, na kterém jsme řešili jednotlivé části projektu, pochopitelně také v anglickém jazyce.

2 Popis odborného zaměření firmy a pracovního zařazení studenta

V této kapitole se seznámíme s firmou, ve které jsem vykonával odbornou praxi, dále pak s pracovním zařazením, oddělením a týmem. Nakonec i s popisem projektu, na který jsem byl přidělen.

2.1 Představení firmy

Tieto patří mezi největší IT společnosti v severní Evropě. Společnost vznikla v roce 1968 s názvem Tietotehdas Oy. Od roku 2009 známe firmu jako Tieto Corporation. Po otevření Ostravské pobočky v roce 2004 se Tieto stalo jedním z největší zaměstnavatelů v oblasti IT v České republice, kde zaměstnává přes 2500 lidí. Dále Tieto působí v téměř 20 zemích.[1] Pobočky firmy můžeme najít například ve Finsku, Švédsku a Indii. Služby, které Tieto nabízí, jsou velmi komplexní a jsou nabízeny jak do veřejného sektoru, tak i do sektoru soukromého. Tieto poskytuje IT služby středním a velkým společnostem z oblasti telekomunikace, výrobního průmyslu, strojírenství a lesnictví, zdravotnictví, sociálních služeb, energetického, ropného a plynárenského průmyslu.[6]

2.2 Pracovní zařazení

Nastoupil jsem na pozici software developera na oddělení CEM, které zaměstnává v Ostravě asi 30 lidí, ve světě pak přes 450 zaměstnanců a zabývá se především technologiemi a oblastmi:

- Tvorba webových aplikací
- Episerver
- Shrarepoint
- Azure
- Genesys

Já jsem konkrétně působil na jednotce OX. Při své stáži jsem pracoval zejména v jazyce C# a s platformou Censhare.

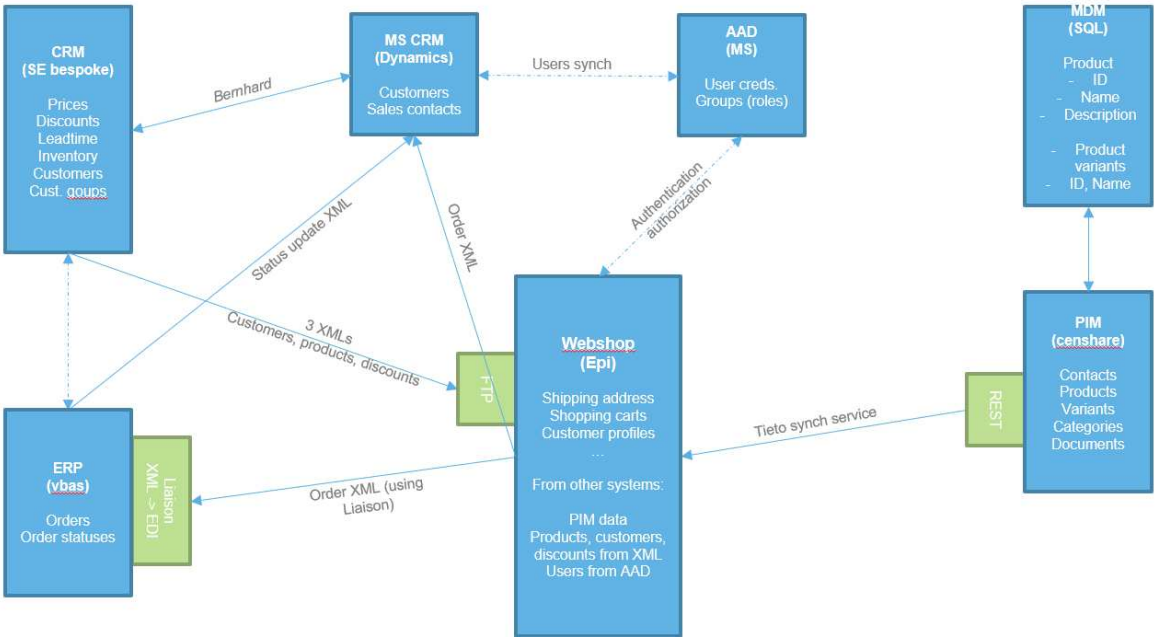
2.3 Popis projektu

V rámci své bakalářské praxe jsem působil na dvou projektech pro německou firmu FläktGroup, která působí zejména na Evropském trhu a zabývá se prodejem klimatizací a ventilací. Popis rozložení technologických modulů firmy zobrazuje Obrázek 1.

První projekt s názvem Webshop rozšiřoval webové stránky, které v minulosti pro FläktGroup vytvářela konkurenční firma Sogeti. Jednalo se tedy o vytvoření fungujícího elektronického obchodu, tak aby produkty, které zatím byly na stránkách v katalogu jen jako neprodejné, mohly být nově i zakoupeny zákazníky.

Firma FläktGroup používala pro správu svých produktů systém Perfion a MDM databázi. Jelikož se rozhodla pro správu produktu začít používat systém od německé společnosti Censhare, v druhém projektu jsme pracovali hlavně s tímto systémem. V projektu nazvaném PIM replacement jsme

pracovali zejména na automatické synchronizaci dat z MDM databáze do Censhare, konfiguraci Censhare, konfigurace Censhare UI, přesunu dat ze stávajícího systému do Censhare, práce s Censhare API, tvorba XSL transformací a synchronizaci dat do elektronického obchodu.



Obrázek 1: Rozložení modulů firmy FläktGroup

Vize firmy FläktGroup byla tedy vytvoření prázdného produktu v MDM databázi a následná automatická synchronizace do Censhare. Tam produkt obalí o všechny potřebné informace, obrázky, dokumenty, varianty produktů, jeho části a překlady textů atd. a po nastavení publikačního statusu je produkt automatiky přenesen na web.

3 Seznam zadaných úkolů a jejich časová náročnost

V rámci praxe jsem pracoval na spoustě úkolů, některé byly krátké a jednoznačné, jiné složité a dlouhé. Celý projekt se v podstatě zabýval systémem Censhare, který byl někdy velmi zrádný, chyběla ucelená a jednoduchá dokumentace a mnohdy jediným řešením bylo využít konzultace pracovníků Censhare. V období svého nástupu, na začátku října, byl projekt v začátcích. Probíhalo dokončování analýzy, grafických návrhů a jiných náležitostí, také nebyla ještě podepsána smlouva se zákazníkem. Bylo to ideální období na přípravu a vyzkoušení si různých oblasti, s kterými bych se na projektu mohl potkat. Především se jednalo o tyto oblasti:

- Práce s MVC v ASP.NET
- Práce s API v C#
- Práce s Entity Frameworkem
- Práce s log4net

Také jsem si krátce v rámci své praxe vyzkoušel, jaké to je testovat software. Po dobu asi **10 dnů** jsem testoval vzhled elektronického obchodu podle předlohy, kterou vytvořili designeři se zákazníkem, a také funkcionalitu a chování podle use case scénářů.

3.1 MDM synchronizace

Vytvoření synchronizačního nástroje mezi MDM databází a Censhare byl v podstatě můj hlavní programátorský úkol. Vyvíjel jsem ho v několika fázích a celková časová náročnost byla okolo **15 dnů.** MDM Databáze je databáze všech produktů a jejich variant, a to jak těch aktivních, tak i těch, které již FläktGroup nevyrábí. Požadavek byl vytvořit všechny produkty a varianty, které mají nastavený status PUSHTOPIM na 1 a zároveň v Censhare neexistují. Dále provádět aktualizaci statusu produktu nebo varianty v Censhare, pokud někdo změní status RETIRED na 1 v MDM databázi.

3.2 Censhare generátor kontaktů

Jednoduchá konzolová aplikace, která pracuje s Censhare API pro vytváření kontaktů. Slouží pro testování synchronizace mezi Censhare a elektronickým obchodem. Celkový čas strávený s tímto nástrojem byl **1 den**.

3.3 XML-CSV překladač

Další nástroj pro usnadnění, tentokrát pro práci s překlady elektronického obchodu. Ty jsou uloženy v XML dokumentu. Protože firma, která zajišťovala překlady do jiných jazyků, neumí pracovat s XML, bylo nutné jim dodat slova k překladu ve formátu CSV. Můj nástroj tedy rozložil XML a převedl ho do formátu CSV a také naopak. Časová náročnost byla zhruba **3 dny.**

3.4 Censhare UI konfigurace

Nejdelší dobu, asi **20 dní** jsem strávil s konfigurací UI Censhare podle požadavků zákazníka. Zde byla nutná především komunikace, neboť jsem pracoval se systémem, který jsem neznal. Součástí bylo např.: konfigurace tabulky, metadata widget, XSL transformace, completness check a další prvky typu widget.

3.5 Přesměrování podle QR kódu

Všechny produkty společnosti FläktGroup mají na sobě QR kód. Po naskenování je možné přesměrování na stránky produktu. Při sestavování URL je produkt vyhledáván v několika zdrojích. Jedním z nich byl právě starý systém Perfion, který byl nahrazen systémem Censhare, tedy i toto vyhledávání bylo nutné předělat. Nejprve bylo potřeba zmapovat, jak funguje sestavování URL a poté ho změnit. Celková časová náročnost byla **6 dnů.**

4 Zvolený postup řešení zadaných úloh

U některých úkolů jsem postup řešení konzultoval se zkušenějšími kolegy, zejména u programovacích úkolů, které vyžadovaly hlubší znalosti. V druhém projektu jsme pracovali se systémem, který nikdo z nás neznal. Pouze jeden můj kolega absolvoval krátké dvoudenní školení na některé funkčnosti Censhare. Jednalo se o první projekt, ve kterém Tieto tento systém používalo, a tak jsme byli v podstatě hozeni do vody a museli jsme se naučit plavat.

4.1 MDM synchronizace

Firma FläktGroup používá pro evidenci všech svých produktů a produktových variant relační databázi obsahující asi 1500 produktů a 137 000 variant. Jsou zde uloženy jak produkty, které jsou zrovna v prodeji, tak produkty, které se již nevyrábí a neprodávají. Synchronizace by měla fungovat podle následujících podmínek firmy FläktGroup:

- Všechny produkty a varianty, které jsou v databázi ale nejsou v Censhare a mají status PUSHTOPIM nastavený na 1, by měly být vytvořeny v Censhare.
- Pokud se změní status produktu nebo varianty v databázi, měl by se upravit i v Censhare.
- Vytvořená varianta musí mít relaci na produkt podle ALPHACODE.
- Synchronizace by měla běžet automaticky, ale musí být možnost ji spustit manuálně.

4.1.1 MDM Databáze

Všechny produkty mají jednoznačný identifikační kód tzv. ALPHACODE a jsou uloženy v tabulce PRODUCT_ALPHA právě s primárním klíčem ALPHACODE. Dále je v tabulce mnoho dalších sloupců. Nás však zajímaly pouze tyto:

1. ALPHACODE – Primární klíč pro produkty, cizí klíč pro varianty.
2. DESCRIPTION – Název produktu.
3. PUSHTOPIM – Status toho, zda má být produkt přenášen do Censhare.
4. RETIRED – Status toho, zda je produkt stále možno koupit, nebo už se nevyrábí.

Příklad takové tabulky můžete vidět v Tabulka 1

	ALPHACODE	DESCRIPTION	PUSHTOPIM	RETIRED
1	ABRA	Air Handling Unit	1	1
2	ACTA	Nozzle duct diffuzer	1	0
3	ACTI	Nozzle duct diffuzer	1	0
4	APAL	Motor single speed	1	0
5	APAT	Motor	1	0
6	APEC	Motor	1	0

Tabulka 1: Ukázka tabulky PRODUCT_ALPHA z MDM databáze firmy FläktGroup

Všechny varianty mají jednoznačný identifikační kód tzv. ONECODE a jsou uloženy v tabulce PRODUCT_MASTER právě s primárním klíčem ONECODE. Dále je v tabulce spousta dalších sloupců. Nás však zajímaly pouze tyto:

1. ONECODE – Primární klíč pro varianty.
2. ALPHACODE – Cizí klíč na tabulku PRODUCT_ALPHA
3. EXCLUDE_FROM_PRICING – Status toho, zda je variantu stále možno koupit, nebo už se nevyrábí

Příklad takové tabulky můžete vidět v Tabulka 2

	ONECODE	ALPHACODE	EXCLUDE_FROM_PRICING
1	SPBA-100-2	SPBA	1
2	SPBA-100-21	SPBA	1
3	SPBA-100-22	SPBA	1
4	SPBA-100-23	SPBA	1
5	SPBA-100-24	SPBA	1
6	SPBA-100-25	SPBA	1

Tabulka 2: Ukázka tabulky PRODUCT_MASTER z MDM databáze firmy FläktGroup

4.1.2 API

Pro vytváření a aktualizaci asset z kódu jsme v celém projektu využívali Censhare REST API. Pro práci s API jsme používali nuget balíček RestSharp. V podstatě bylo možné voláním API provést v Censhare cokoliv. To, co se stane, určovala XSL transformace, která je součástí Censhare, ta se volá v API base URL a v těle se předává struktura ve formátu XML. Využil jsem tyto transformace:

- Na vytváření – flaktgroup:transformation.create-asset-from-rest
- Na získání dat – flaktgroup:transformation.get-all-asset
- Na aktualizaci produktu – flaktgroup:update-asset-string

Příklad použití API z kódu vytvářející asset typu produkt je ve Zdrojový kód 1.

```
RestClient client = new RestClient(ip + "ws/rest/service/assets/asset;Censhare:re-
source-key=flaktgroup:transformation.create-asset-from-rest/transform;key=flakt-
group:transformation.create-asset-from-rest")
{
      Authenticator = new HttpBasicAuthenticator(username, password)
};

var request = new RestRequest(Method.POST);

string text = "<asset name="TESTOVACI PRODUKT" type="product."></asset>";

request.AddParameter("text/xml", text, ParameterType.RequestBody);

IRestResponse response = client.Execute(request);
```

Zdrojový kód 1: Vytváření asset voláním API pomocí knihovny RestSharp

4.1.3 Entity Framework

Jako ORM jsem použil Entity Framework. Přístupem Database First jsem vytvořil EDMX model podle zálohy databáze, který nám poslala firma FläktGroup. Nemohli jsme totiž pracovat přímo s originální databází firmy, neboť ta byla nepřístupná ze sítě, ve které jsem pracoval. Nutnou součástí správného fungování synchronizace bylo povoleni cesty skrz firewall z místa, kde bude program nasazen, do místa, kde leží MDM databáze. Potom už stačilo vyžádat si přihlašovací údaje a změnit connection string.

4.1.4 Logika programu

Nejprve bylo třeba načíst data ze Censhare a udělat z nich objekty, poté načíst data z databáze a udělat z nich objekty, dále porovnat je mezi sebou, vytvořit takové produkty a varianty, které jsou v databázi ale nejsou v Censhare, a nakonec aktualizovat takové produkty, u kterých se v databázi změnil jejich status RETIRED na 1.

Pro snadné porovnávání jsem si vytvořil třídy Product a Variant, obsahující vlastnosti alphacode, status, u produktu navíc description a pushtopim, u varianty navíc onecode.

Dále jsem si vytvořil třídu DatabaseRepository obstarávající operace nad databází. Konstruktor inicializoval model databáze pomocí Entity Framework. Funkce GetAllProducts iteruje kolekci DbSet tabulky PRODUCT_ALPHA a v případě, že má produkt atribut PUSHTOPIM nastaven na 1, vytvoří instanci třídy Product, naplní jej daty z databáze a přidá jej do seznamu produktů, který je výstupem funkce. Stejným způsobem funguje funkce GetAllVariants. Ta navíc iteruje v seznamu načtených produktů, tak aby se z databáze načetly pouze varianty již načtených produktů, ne žádné jiné. Tuto funkci můžete vidět ve Zdrojový kód 2.

```
public IEnumerable<Variant> GetAllVariants()
{
	IList<Variant> variants = new List<Variant>();

	foreach (var item in db.PRODUCT_MASTER)
	{
		foreach(var i in products)
		{
			if (i.AlphaCode.Trim() == item.ALPHACODE.Trim())
			{
				variants.Add(new Variant {
					AlphaCode = item.ALPHACODE.Trim(),
					Retired = item.RETIRED,
					OneCode = item.ONECODE.Trim()
				});
			}
		}
	}
	return variants;
}
```

Zdrojový kód 2: Ukázka vytváření Variant nad daty z databáze

Další třída CenshareRepository obstarává operace nad Censhare. V konstruktoru se načítají přihlašovací údaje a IP adresa serveru. Funkce GetAssetId vrátí asset ID podle alphacode tak, že iteruje kolekci načtených produktů a slouží pro navazování relací na produkty při vytváření variant.

Funkce GetAllAssetByType přijme v parametru typ asset. Funkce volá API pomocí třídy RestClient. Dále vytváří instanci třídy RestRequest s metodou POST. Jako parametr mu předává sestavený textový řetězec funkcí AddParameter. Sestavený požadavek je pak předán funkci Execute, která provede požadované volání API a vrátí odpověď, která je uložena jako rozhraní IRestResponce a je z ní vytvořen XML dokument.

XML dokument je dále zpracován ve funkcích GetAllProducts a GetAllVariants, které pomocí procházení XML uzlů vytvoří instanci objektu Product a Variant, naplní ho daty z XML dokumentu, které jsou jako atributy uzlů a hotové objekty přidají do seznamu, který je výstupem funkcí.

Vytváření produktů a variant v Censhare obstarávají funkce CreateProductAsset a CreateVariantAsset a je realizováno rovněž pomocí RestSharp stejným způsobem. Vytvoří se RestClient s příslušným URL v parametru, nastaví autentizační údaje, vytvoří se textový řetězec s daty produktu nebo variant předaného v parametru funkce, vytvořený textový řetězec je předán jako parametr RestRequest a ten je zavolán funkcí Execute.

Poslední funkcionalita se týká aktualizace statusu produktu v Censhare. Ta je realizována funkcí UpdateAssetStatus na stejném principu použití knihovny RestSharp. Funkci je předáno identifikační číslo asset, jehož status má být změněn. Ukázku funkce můžete vidět v Zdrojový kód 3.

```
public bool UpdateAssetsStatus(int assetId)
{
       RestClient client = new RestClient(ip+ "ws/rest/service/assets/asset/id/" +
       assetId+ "/transform;key=flaktgroup:update-asset-string;feature-name=flakt-
       group:product-status;feature-value=retired")
       {
              Authenticator = new HttpBasicAuthenticator(username, password)
       };

       var request = new RestRequest(Method.GET);

       IRestResponse response = client.Execute(request);
}
```

Zdrojový kód 3: Ukázka volání metody pro aktualizaci statusu asset

Poslední třída MDMSynchronizer v sobě skrývá logiku porovnávání objektů. Nejprve jsou načteny seznamy produktů a variant z Censhare a z databáze. Zde jsem převážně využil jazyk LINQ pro operace nad kolekcemi.[4] V případě, že je nalezen produkt, který není v kolekci z Censhare, ale je v kolekci z databáze, je funkcí CreateProductAsset vytvořen. V případě, že je nalezen produkt, jehož status je jiný u produktu z databáze, než u produktu z Censhare je aktualizován pomocí funkce UpdateAssetStatus. Obdobné porovnání se následně provádí i pro varianty. Ukázku porovnávání objektů můžete vidět v Zdrojový kód 4.

```
foreach (var item in allProductsFromDb)
{
       if (item.PushToPim && !allProductsFromCenshare.Any(x => x.AlphaCode ==
       item.AlphaCode))
       {
              CenshareRepository.CreateProductAsset(item);
       }

       if (item.Retired && allProductsFromCenshare.Any(x => x.AlphaCode ==
       item.AlphaCode))
       {
              if (allProductsFromCenshare.Where(x => x.AlphaCode == item.Alpha-
              Code).FirstOrDefault().Retired == false)
             {
                    item.AssetId = allProductsFromCenshare.SingleOrDefault(x =>
                    x.AlphaCode == item.AlphaCode).AssetId;
                    CenshareRepository.UpdateAssetsStatus(item.AssetId);
             }
       }
}
```

Zdrojový kód 4: Ukázka porovnávání produktů z Censhare a z databáze

4.1.5 Log

Jelikož synchronizace běží automaticky, bylo nutné zakomponovat do kódu sekci try-catch, aby aplikace nepadala, ale v případě nějaké chyby byla zaznamenána do souboru a běžela dál. Pro zaznamenávání průběhu programu jsem použil externí knihovnu log4net. Každý den se vytvoří nový textový soubor a zaznamenávají se do něj všechny události. V případě chyby (špatné přihlašovací údaje, nedostupný server, nedostupná databáze apod.) jsou detaily zaznamenány v textovém souboru příslušného dne.

4.1.6 Nasazení

Finální rozhodnutí o tom, kde bude kód nasazen, padlo po dlouhé diskuzi. Nakonec jsme se rozhodli, že bude součástí solution v Episerver, kde jsou i webové stánky a e-shop. Bude přidán jako externí knihovna proto, aby nebyla závislá na webových stránkách, ale dala se libovolně přenést a nasadit jinde a bude využívat funkci tzv. schedule job, který se dá spouštět jak manuálně, tak lze jeho spouštění nastavit automaticky. Navíc je to uživatelsky přehledné, byť to nemá s Episerver nic společného.

4.2 Censhare generátor kontaktů

Součástí projektu byl nástroj pro synchronizaci mezi Censhare, kde jsou uloženy všechny informace o produktech, kontaktech a dokumentech a vytvářeným elektronickým obchodem. Bylo nutné věnovat spoustu času testování synchronizace, neboť její správné fungování bylo stěžejní pro bezproblémové přelívání dat z Censhare do elektronického obchodu.

Takovou testovací aplikací byl i mnou vytvářený generátor kontaktů v Censhare. Realizoval jsem ji jako konzolovou aplikaci. Pro tuto příležitost jsem si vytvořil tři třídy. Třidu Contact, která obsahovala vlastnosti všech vytvářených dat, jako například: Name, Country, Company atd. Třídu DataStorage, která uchovávala seznamy všech testovacích dat, která byly uloženy v XML dokumentu Configuration.xml a měla implementovanou funkci indexer pro naplnění seznamů daty z XML dokumentu Configuration.xml. A třídu Run, která obsahovala logiku programu.

Nejprve se po vytvoření instance třídy volá v konstruktoru funkce InicializeDataStorege, která vtvoří instanci třídy DataStorage a načte soubor Configuration.xml. Pomocí funkce SelectSingleNode z něj vytáhne hodnotu atributů toCreate, který představuje počet kontaktů, které se mají vytvořit a pomocí funkce FillDataStorage naplní instanci třídy DataStorage za pomocí funkce indexer. Ukázku zmíněné funkce můžete vidět v Zdrojový kód 5.

```
private void FillDataStorage(XmlDocument xDoc, string name)
{
	XmlNode node = xDoc.SelectSingleNode(@"/set/contacts/" + name);

	foreach (XmlNode nn in node.ChildNodes)
	{
		data[name] = nn.InnerText;
	}
}
```

Zdrojový kód 5: Ukázka zpracování XML dokumentu

Nakonec se v konstruktoru spustí cyklus a v každé iteraci se volá funkce CreateContactAsset. Tato funkce nejprve volá funkci GetRandomContact, která vytvoří instanci třídy Contact a přiřadí jí náhodná data z objektu DataStorage a poté vytváří za pomocí RestSharp kontakt v Censhare voláním jejich API. Více o práci s API nalezete v kapitole 4.1.2 Výstupem jsou pak vytvořené asset typu person.flakgroupContact. Více o tom, co je to asset, je popsáno v kapitole 4.4.1

4.3 XML-CSV překladač

Nástroj pro překlad z XML do CSV a zpět jsem realizoval pomocí dvou konzolových aplikací napsaných v jazyce C#.

4.3.1 Konverze do CSV

Aplikace pro konverzi z XML do CSV bylo celkem jednoduché vytvořit. Vstupním souborem byl XML dokument přiložený do složky bin. Výstupem byl CSV dokument, který obsahoval dva sloupce. Do prvního sloupce jsem ukládal klíč, v podstatě cestu skrz XML uzly. Příklad takového klíče:

„languages/language/pages/productPage/chooseProductSection/chooseProduct/. Každé slovo mezi lomítky reprezentuje název uzlu. Do druhého sloupce se ukládá textová hodnota XML elementu.

V první fázi se provádí rekurzivní funkce GetAllNodes, která projede celý XML dokument a uloží do seznamu typu List jedinečné uzly, jestliže nemá dalšího potomka. V opačném případě se volá rekurzivní funkce GetAllNodes. V druhé fázi používám třídy FileStream pro přístup k souboru a StreamWriter pro zápis do souboru. V tomto případě bylo důležité ošetřit kódování, neboť se v překládaném textu vyskytovaly švédské znaky např.: Ö, nebo Ä, proto jsem použil kódování UTF-8. Postupně jsem projel dříve připravený seznam uzlů a vytvořil pro každý záznam cestu funkci GetXPath. Příklad takové funkce je Zdrojový kód 6. Vytvořená cesta byla uložena do prvního sloupce a příslušná hodnota XML uzlu do sloupce druhého.

```
private static string GetXPath(XmlNode node)
{
	string result = "";

	while (node.ParentNode != null && node.ParentNode.NodeType !=XmlNodeType.Do-
	cument)
	{
		node = node.ParentNode;
		if (result == "")
		{
			result = node.Name;
		 }
		 else
		{
			result = node.Name + "/" + result;
		}
	}
	return result;
}
```

Zdrojový kód 6: Ukázka získání cesty po uzlech předků

4.3.2 Konverze do XML

O něco náročnější byla zpětná konverze do XML. Potýkali jsme se s problémy, kdy postupně přibývaly další slova pro překlad a bylo nutné spojovat CSV soubory. Proto jsme se rozhodli vytvořit jeden CSV soubor v sdíleném adresáři, ke kterému mají přístup všichni vývojáři a mohou do něj přidávat klíče a hodnoty pro překlad. Jednou za čas se soubor poslal firmě, která překládá anglické názvy do švédštiny. Vstupem byl CSV dokument a výstupem XML dokument.

V první fázi jsem načetl data z CSV souboru do slovníků pomocí funkce GetDataFromCsv. Funkce využívá třídu StreamReader pro čtení ze souboru, rozkládá řádek ze souboru do dvou textových řetězců pomocí funkce String.Split a je uložen do slovníku, který je výstupem funkce. Ukázku práce se streamy můžete vidět v Zdrojový kód 7.

```
private static Dictionary<string, string> GetDataFromCsv(string path)
{
	Dictionary<string, string> result = new Dictionary<string, string>();

	using (StreamReader sr = new StreamReader(path, Encoding.UTF8))
	{
		while (!sr.EndOfStream)
		{
			string[] toks = sr.ReadLine().Split(';');

			result.Add(toks[0], toks[1]);
		}
	}
	return result;
}
```

Zdrojový kód 7: Ukázka práce se streamy

V druhé fázi bylo třeba klíče ve slovníku seřadit podle abecedy, vytvořit XML dokument, přidat jeho kořenový uzel, přiřadit deklaraci a přiřadit atribut obsahující název jazyka XML dokumentu kořenovému uzlu.

Dále program iteroval všechny položky ve slovníku tak, že rozložil klíč na jednotlivé názvy uzlů, ty pak iteroval a postupně vytvářel, pokud již neexistovaly. Pro tuto kontrolu jsem si vytvořil seznam již vytvořených uzlů. Ve chvíli, kdy program iteroval poslední název v klíči, přiřadil uzlu text pomocí funkce `XmlNode.AppendChild`. Program si po celou dobu držel aktuální rodičovský uzel, na který se nově vtvořený uzel napojoval. Tím byla zaručena správnost a konzistentnost výsledné struktury XML dokumentu.

4.4 Censhare UI konfigurace

4.4.1 O platformě Censhare

Censhare je komerční platforma ve formě ECMS od německého výrobce software Censhare AG.[5] Je to v podstatě redakční systém, který je schopen pracovat s velkým množstvím dat a vazeb. Firma Censhare poskytuje svou platformu firmám jako například: Mc Donald´s, Hearst Magazine UK, GoPro, BMW, Oriflame, Toyota a další.[2]

Základním pojmem pro jakékoliv místo v Censhare uchovávající nějaké informace je asset. Každý asset má svůj typ například: produkt, dokument, kontakt, nebo třeba video. Dále má svůj název a doménové umístění. Asset dále může obsahovat tzv. feature. Feature je úložiště, kde se v Censhare ukládají data, například je zde možno uložit jméno produktu, váhu nebo třeba emailovou adresu kontaktu. Dále asset nese informace o relacích, jak rodičovských například v jaké kategorii je produkt, tak o potomcích například v jakých zemích se produkt prodává.

4.4.2 Náplň práce

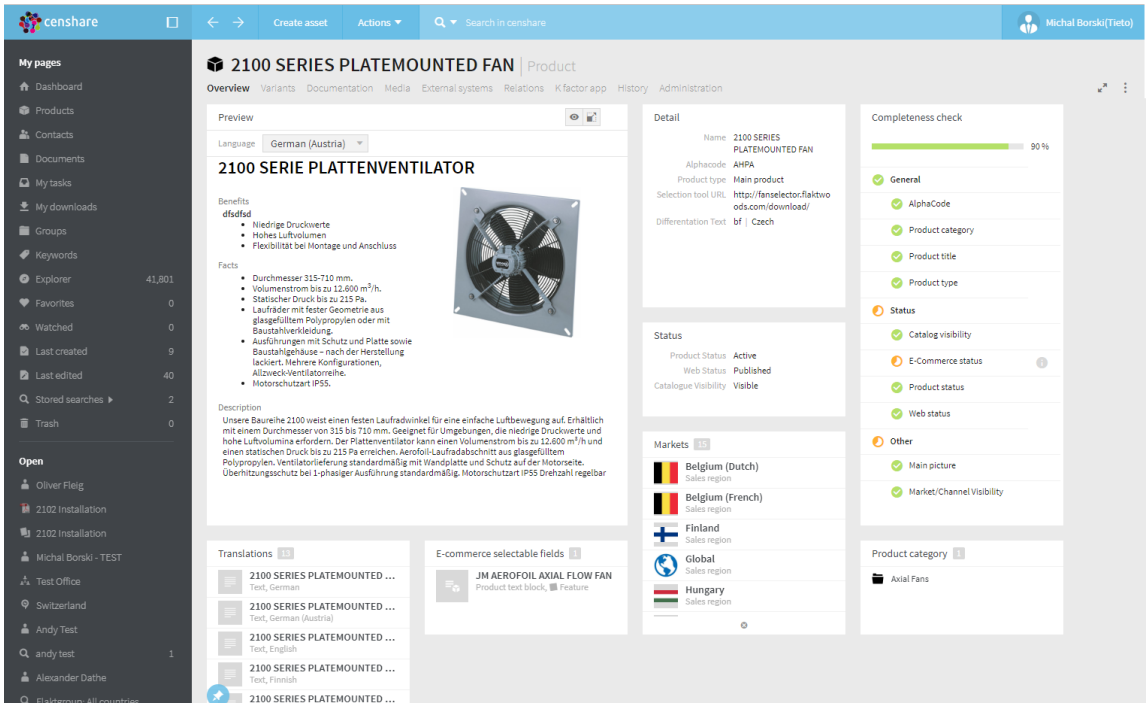
V první fázi zákazník vytvořil požadavky pro to, jaké pracovní plochy (dále workspace) mají být nakonfigurovány a jaké informace a relace mají být zobrazeny. Nutno říct, že jsem s tímto systémem nikdy nepracoval, a proto vůbec pochopit, jak funguje, bylo těžké. Na pomoc jsem měl velmi slabou dokumentaci firmy Censhare a tři konzultanty Marca Flatchera z Britské pobočky a Martina Friberga

z Švédské pobočky a Tomase Martiniho. Byla to velmi investigativní práce. Snažil jsem se rozebírat existující prvky typu widget, zkoušet různé postupy, ale bylo to velmi složité. Životně důležitá byly schopnost komunikace v anglickém jazyce, v podstatě každý den jsem absolvoval schůzky se zákazníkem, se kterým jsme probírali jednotlivou podobu jeho požadavků a řešili věci, které nemohly být vytvořeny tak, jak si představoval. Dále jsem měl spoustu schůzek, kde jsem prezentoval výsledky své práce. Zde narážím na velkou nestandardnost studentské praxe. Není úplně běžné, a zřejmě by to tak nemělo být , aby student prezentoval před zákazníkem výsledky práce celého týmu. Nicméně jsem to zvládl a od té doby se to stalo běžnou rutinou.

Postupem času jsem své dílo vylepšoval, naučil jsem se třeba některou novou funkčnost a dobral jsem se výsledku, s kterým byl zákazník spokojen. Nutno říct, že z některých svých požadavků zákazník nakonec slevil, protože jsme nenašli způsob, jak jej uspokojit, podle toho, co Censhare umožnoval.

4.4.3 Workspace a widget

Konfigurace workspace a widget je spojená s tím, jak uživatel vidí Censhare po přihlášení. Workspace je pracovní plocha rozdílná pro každý typ asset. Tím je myšleno, že všechny asset typu produkt budou mít stejnou UI konfiguraci, ale asset typu kontakt už budou mít jinou. Widget je pak jednotlivý čtvereček. Příklad takového již vytvořeného workspace můžete vidět na Obrázek 2.



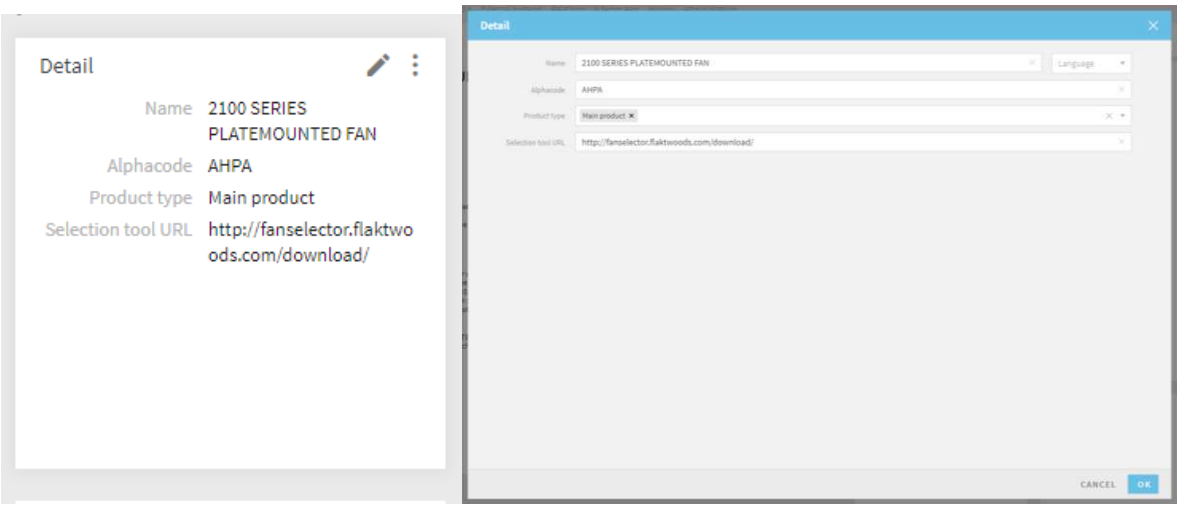
Obrázek 2: Ukázka produkt workspace

Seznam workspace, které bylo potřeba nakonfigurovat pro potřeby zákazníka:

- Produkt a produktová varianta
- Kategorie produktu
- Kontakt
- Kancelář a Země
- Dokument set a Dokument

4.4.3.1 Metadata widget

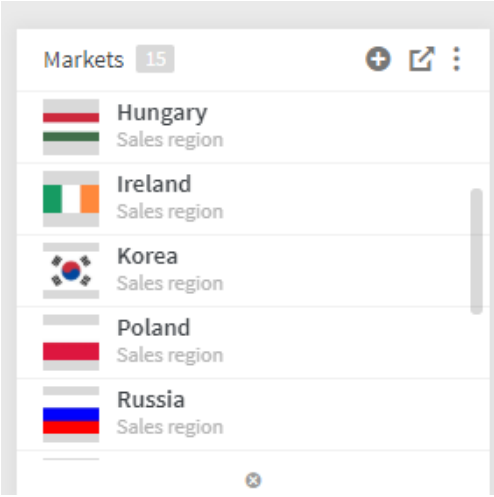
Jedná se o widget obsahující textová data. Konfiguruje se ve dvou režimech, pro čtení a pro zápis. Pracuje se s ním pomocí HTML značek. V atributu s názvem *property* se uvádí zdroj dat, která se mají zobrazit. Příklad takového widget můžete vidět na Obrázek 3.



Obrázek 3: Ukázka metadata widget pro čtení a úpravu

4.4.3.2 Relation widget

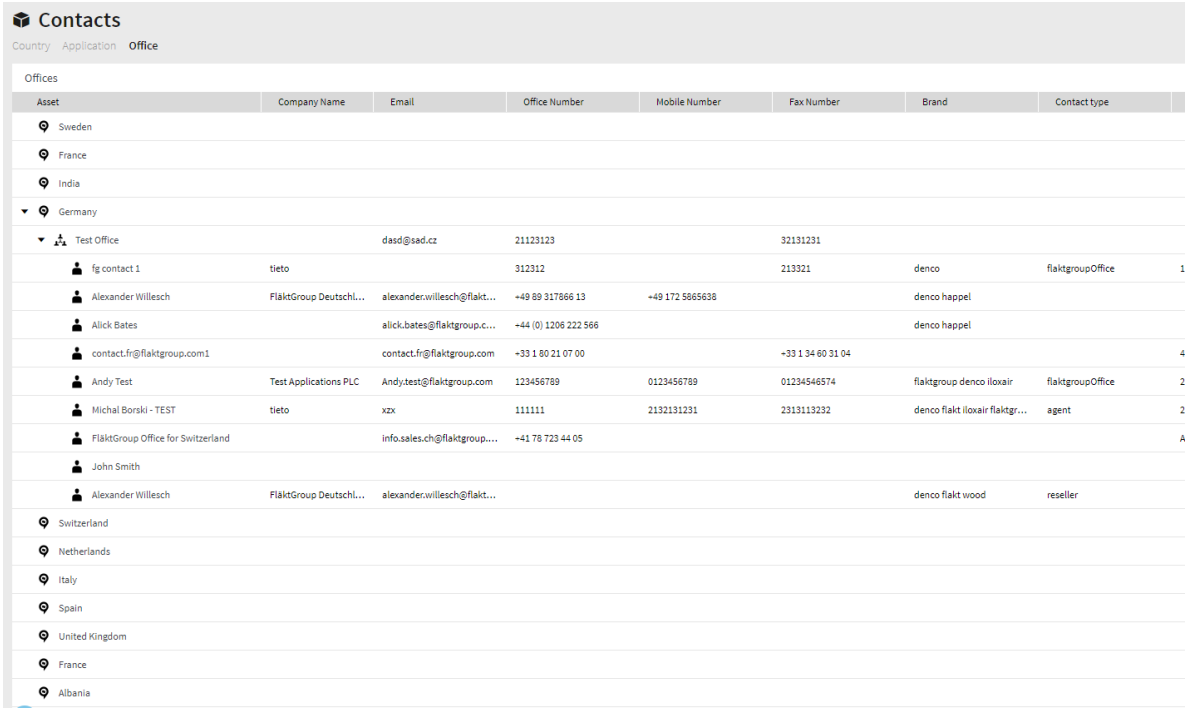
Konfigurace relačního prvku typu widget probíhala tak, že se v první fázi vytvoří relace, přiřadí se ji rodičovské relace a relace pro potomky a poté se nakonfiguruje widget, aby zobrazoval danou relaci. Ukázku relačního widget můžete vidět na Obrázek 4.



Obrázek 4: Ukázka relačního prvku typu widget

4.4.3.3 Table widget

Nejprve bylo potřeba vytvořit asset typu tabulka, poté asset typu column. Do nich umístit informaci o tom jaká data mají zobrazovat, buď formou xPath, nebo XSL transformace. Dále se přiřadí sloupce tabulce jako relace. To, co se bude v tabulce zobrazovat definuje XSL transformace root query, například všechny země. Child query definuje co se bude zobrazovat pod zeměmi, například všechny kanceláře a kontakty. Ukázka tabulky je na Obrázek 5.



Obrázek 5: Ukázka tabulkového prvku typu widget

XSL transformaci, zobrazující všechny kanceláře a kontakty, můžete vidět na Obrázek 6.

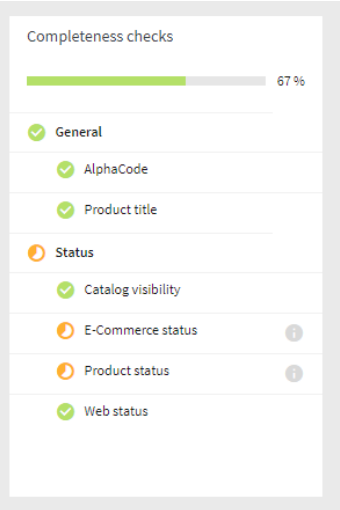
Text Editor

```
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xi="http://www.w3.org/2001/XInclude"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:cs="http://www.censhare.com/xml/3.0.0/xpath-functions">

  <!-- Search from the current asset all child user.media. related picture assets sorted by the order of the media relation -->
  <xsl:template match="/">
    <query>
      <or>
        <relation direction="parent" type="user.country-department.">
          <target>
            <and>
              <condition name="censhare:asset.id" op="=" value="{asset/@id}"/>
            </and>
          </target>
        </relation>

        <relation direction="parent" type="user.department-employee.">
          <target>
            <and>
              <condition name="censhare:asset.id" op="=" value="{asset/@id}"/>
            </and>
          </target>
        </relation>
      </or>
    </query>
  </xsl:template>

</xsl:stylesheet>
```

Obrázek 6: Ukázka XSL transformace zobrazující všechny kanceláře a v nich kontakty

4.4.3.4 Competeness check widget

Completeness check definuje v jaké fázi jsou data produktu, zda jsou nastavena nebo ne. V první fázi se vytvoří asset typu quality gate sequence, tomu se přiřadí asset typu quality gate obsahující už jednotlivé assety typu check, které ověřují, zda je nastaven například aslphacode. Ukázku takového prvku typu widget můžete vidět na Obrázek 7.



Obrázek 7: Ukázka prvku completeness check

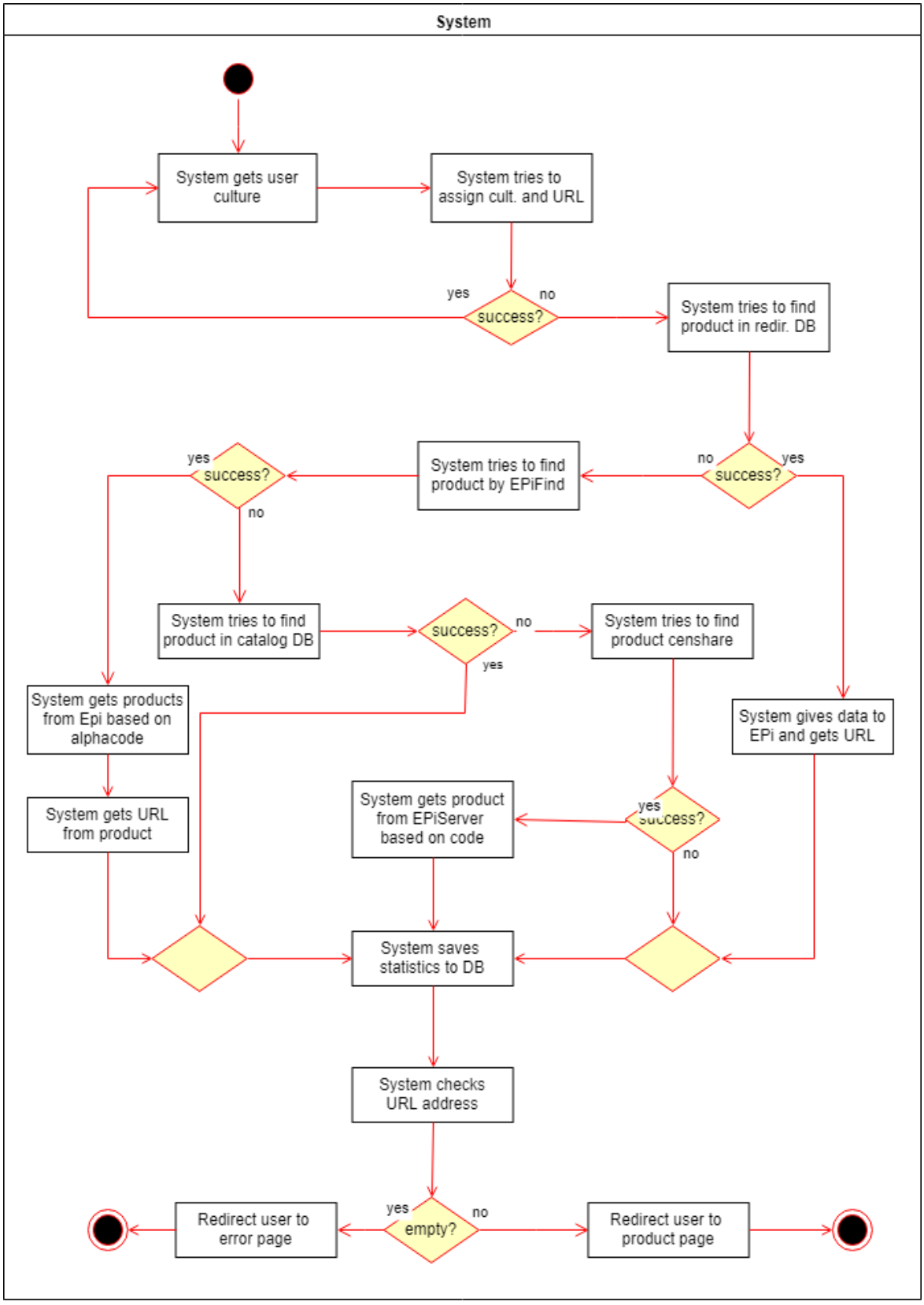
4.4.4 Služební cesta do Mnichova

V rámci své praxe jsem absolvoval dvoudenní workshop v centrále firmy Censhare v německém Mnichově. Zde jsem se sešel, spolu se svými dvěma kolegy z firmy Tieto, se zákazníkem z firmy Fläkt-Group a konzultanty z Censhare. Setkání z očí do očí byla zase nová zkušenost z pohledu komunikace. Je rozdíl mluvit s člověkem přes Skype, anebo se mu dívat do očí. Zde jsme prohloubili naše znalosti a pochopili o něco více, jak Censhare funguje do detailu a jak je to komplexní a promyšlený systém. Také jsme měli možnost, tím že byl přítomen i zákazník, vyjasnit si všechny jeho požadavky, ohledně doménového rozložení, UI konfigurace a dalších bodů.

4.5 Přesměrování podle QR kódu

Pro tento úkol byla potřeba pouze přepracovat a doplnit existující kód sestavování URL produktu. V první fázi bylo potřeba zmapovat, jak sestavování URL funguje. Vytvořil jsem aktivitní diagram, který můžete vidět na Obrázek 8.[3]

Program nejprve zkouší najít již použité přesměrování v databázi přesměrování. Pokud nic nenajde zkouší najít produkt podle alphacode pomocí integrovaného vyhledávacího modulu EpiFind, který je součástí Episerveru. Když neuspěje, zkouší najít produkt v databázovém katalogu. Pokud stále produkt nebyl nalezen, přichází vyhledání v systému Perfion, a právě tahle část se musela změnit, neboť systém Perfion byl nahrazen Censhare. Vytáhl jsem si tedy všechny produkty z Censhare pomocí API za použití knihovny RestSharp, projel vrácený XML dokument a v případě, že byl produkt podle alphacode nalezen, je sestavena URL.



Obrázek 8:Aktivitní diagram zobrazující sestavování URL QR knihovny

5 Zhodnocení znalostí a dovedností

5.1 Uplatněné teoretické a praktické znalosti a dovednosti získané v průběhu studia

Zhruba polovinu odborné praxe jsem strávil programováním. Využil jsem spoustu praktických dovedností a znalostí získaných v průběhu studia na vysoké škole. Nejdůležitější byla znalost programovacího jazyku C# a platformy .NET v kombinaci s objektově orientovaným přístupem programování. Tyto znalosti jsem nabyl zejména z předmětů Programovací jazyky 1 a 2 a také Architektura .NET. Konkrétně jsem pak často využíval znalosti pro práci s XML. Dále jsem využil jazyk SQL pro manipulaci s databází, který jsem si osvojil v předmětu Úvod do databázových systémů. Dále jsem se setkal s prvky jazyka UML, který jsem se naučil v předmětu Úvod do softwarového inženýrství, konkrétně jsem konstruoval aktivitní diagram mapující práci programu.

V druhé části jsem pracoval především na konfiguraci UI Censhare. Zde byla velmi podstatná hlavně schopnost komunikace v anglickém jazyce. Anglický jazyk jsem měl 3 semestry na vysoké škole, takže se do jisté míry dá říct, že i tuhle znalost jsem využil.

5.2 Znalosti a dovednosti scházející v průběhu odborné praxe

Mezi znalosti a dovednosti, které mi chyběly při odborné praxi bylo na prvním místě práce s Git. Z počátku to bylo pro mě velmi zmatené pochopit princip toho, jakým způsobem Git funguje. Všechny pojmy jako commit, branch, fetch, push a merge byly pro mě doposud neznámé. Myslím si, že práce s Git by měla být zahrnuta do některého z programovacích předmětů.

Další znalost, která mi scházela, byla práce s REST API. V průběhu praxe jsem pracoval jak na vytváření, tak zejména na jeho volání. Používání knihoven jako například RestSharp by bylo vhodné zařadit do některého předmětu zabývající se programovacími jazyky.

6 Závěr

Díky absolvování odborné praxe ve firmě Tieto jsem se posunul o kus dopředu. Vyzkoušel jsem si, jaké to je být zaměstnán v tak velké firmě, výrazně jsem zlepšil své komunikační dovednosti, a to zejména v anglickém jazyce, nabyl jsem spoustu zkušeností s programovacími technikami, ale především jsem mohl využít své znalosti z vysoké školy v praxi na projektech, které nejsou jen práce do šuplíku, ale mají reálnou hodnotu a bude je někdo využívat.

Každý produkt nebo produktová varianta vytvořená v MDM databázi firmy FläktGroup je přenesena do Censhare mnou vytvořenou synchronizační aplikací. A každý zaměstnanec firmy FläktGroup, který se bude pohybovat v Censhare, bude používat mnou vytvořené workspace. V podstatě má práce stojí za vším, co uvidí a s čím bude pracovat.

Celkově hodnotím praxi velmi kladně. Ze strany kolegů i nadřízených jsem celou dobu cítil respekt k tomu, co dělám a důvěru například v komunikaci se zákazníkem, i když jsem byl stále student.

Literatura

1. Tieto Czech s.r.o. [online]. Ostrava, 2019 [cit. 2019-04-08]. Dostupné z: https://www.tieto.com/cz/about-us/our-company/. Bakalářská práce. VŠB-TUO.

2. Censhare [online]. Ostrava, 2019 [cit. 2019-04-08]. Dostupné z: https://www.Censhare.com/us/client-success/clients. Bakalářská práce. VŠB-TUO.

3. KANISOVÁ, Jana a Miroslav MÜLLER. UML srozumitelně. Praha: Computer Press, 2006. ISBN 8025110834.

4. LINQ tutorial [online]. [cit. 2019-04-08]. Dostupné z: https://www.tutorialspoint.com/linq/

5. Censhare wikipedia.org. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-08]. Dostupné z: https://en.wikipedia.org/wiki/Censhare

6. Tieto wikipida.org. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-08]. Dostupné z: https://en.wikipedia.org/wiki/Tieto